CONSTRUCTING CARMICHAEL NUMBERS THROUGH IMPROVED SUBSET-PRODUCT ALGORITHMS

W.R. ALFORD, JON GRANTHAM, STEVEN HAYMAN, AND ANDREW SHALLUE

Abstract. We have constructed a Carmichael number with 10,333,229,505 prime factors, and have also constructed Carmichael numbers with k prime factors for every k between 3 and 19,565,220. These computations are the product of implementations of two new algorithms for the subset product problem that exploit the non-uniform distribution of primes p with the property that $p-1$ divides a highly composite Λ.

1. Introduction

A Carmichael number n is a composite integer that is a base-a Fermat pseudoprime for all a with $\gcd(a,n)=1$. However, constructions of Carmichael numbers often rely on the following equivalent definition.

Definition 1.1 (Korselt condition)**.** A positive integer n is a Carmichael number if it is composite, squarefree, and has the property that $p-1 \mid n-1$ for all primes p dividing n.

Our goal is to construct Carmichael numbers with a very large number of prime factors. The construction we will use is due to Erdős [4] and has been a popular method since 1992 [18].

Erdős Construction:

(1) Choose $\Lambda = \prod_{i=1}^{r} q_i^{h_i}$ where $q_1 \dots q_r$ are the first r primes in order and the h_i are all at least 1 and non-increasing.
(2) Construct the set $\mathcal{P} = \{p \text{ prime } : \ p-1 \mid \Lambda, p \nmid \Lambda\}$
(3) Construct Carmichael n as a product of primes in $\mathcal{P}$ in one of two ways:
   (a) Find a subset $\mathcal{S}$ of $\mathcal{P}$ such that
   $$\prod_{p\in\mathcal{S}} p \equiv 1 \mod \Lambda \ .$$
   Then by Definition 1.1 $n = \prod_{p\in\mathcal{S}} p$ is Carmichael.
   (b) Alternatively, let $b \equiv \prod_{p\in\mathcal{P}} p \mod \Lambda$ and find a subset $\mathcal{T}$ of $\mathcal{P}$ such that
   $$\prod_{p\in\mathcal{T}} p \equiv b \mod \Lambda \ .$$
   Then $n = \prod_{p\in\mathcal{P}\setminus\mathcal{T}} p$ is Carmichael.

2010 *Mathematics Subject Classification.* Primary 11Y16.
Key words and phrases. Subset sum, Carmichael numbers.
Research supported by an Illinois Wesleyan University grant.
W.R. Alford passed away in 2003.

This notation for Λ will be fixed throughout, along with b as the product modulo Λ of all primes in $\mathcal{P}$. Additionally, we will use Q_i to represent $q_i^{h_i}$ and fix the ordering so that $q_1 = 2$, $q_2 = 3$ and so on.

Choosing a good Λ is something of an art, seeing as how we want a number that is small and yet has many divisors. One possibility (taken as a starting point by the authors in [9]) is to choose Λ to be highly composite [13]. We do not insist upon it here, instead relying on the condition that $h_i \leq r/i$ for $1 \leq i \leq r$ in order to prove bounds on running times. In practice, an excellent tool for choosing Λ is the following function $K(\Lambda)$ from [9] that returns an estimate of $|\mathcal{P}|$.

$$K(\Lambda) = \left\lfloor \frac{\Lambda}{\phi(\Lambda)\log\sqrt{2\Lambda}} \prod_{i=1}^{r} \left(h_i + \frac{q_i - 2}{q_i - 1} \right) \right\rfloor$$

In terms of constructing $\mathcal{P}$, all divisors d of Λ are checked to see if $n = d + 1$ is prime. Primality proofs are easy since we are given the factorization of $n - 1$. They are also necessary; the second author has found pseudoprimes while testing primality using randomized algorithms.

Löh and Niebuhr noted that step (2) is by far the most costly. Nevertheless, the improvements we present will be to step (3), seeing as how the subset product problem in such a dense set of instances is fertile ground for algorithmic advancement. In addition, step (2) is easily parallelized while step (3) is not, and step (2) requires almost no memory while the space requirement of most subset product algorithms is high.

Our new contribution involves improvements to existing literature on a broad array of fronts. One new algorithm combines a series of smaller Carmichael numbers into larger ones, enabling quick construction of Carmichaels with a variety of factor counts. Another new algorithm incorporates ideas from [9] and [5] to achieve a randomized method that solves the subset product problem using subexponential time and space (in fact, sublinear in N). Computationally this has resulted in Carmichael numbers with 10333229505 prime factors and with k prime factors for every k between 3 and 19565220.

One key insight is that elements of $\mathcal{P}$ are not distributed uniformly in $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$, and this non-uniformity can be exploited. Another is that it is useful to make products that get successfully closer to the identity in $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$. We will use two such functions for our algorithms, the first of which comes from [9].

Definition 1.2. Let $a \in (\mathbb{Z}/\Lambda\mathbb{Z})^\times$. Then $\omega(a)$ is an integer between 0 and r defined by

$$\omega(a) = \max_{a \bmod Q_i \neq 1} i$$

unless $a \bmod Q_i = 1$ for all $1 \leq i \leq r$, in which case $\omega(a) = 0$.

Definition 1.3. Let $a \in (\mathbb{Z}/\Lambda\mathbb{Z})^\times$. Then $\bar{\omega}(a)$ is an integer between 0 and r defined by

$$\bar{\omega}(a) = \min_{a \bmod Q_i \neq 1} i$$

unless $a \bmod Q_i = 1$ for all $1 \leq i \leq r$, in which case $\bar{\omega}(a) = r + 1$.

Formally, the subset product problem over an abelian group is defined as follows.

Definition 1.4. Let G be an abelian group written multiplicatively with $(a_1, \ldots, a_N, b)$ a list of elements of G. Then the subset product problem $(a_1, \ldots, a_N, b)$ is to determine a sublist of the a_i that product to b in G.

In the Erdős construction G is typically $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$, but we will also consider subset product problems on subgroups of $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$.

The subset product problem is NP-hard, but the difficulty of a particular instance can vary depending on its density. The hardest problems are those of density 1.

Definition 1.5. The density of a subset product problem $(a_1, \ldots, a_N, b)$ is

$$\frac{N}{\log_2 |G|} \ .$$

Problems arising from the Erdős construction will typically have density much larger than 1, in fact closer to $O(N/\log N)$. We thus expect algorithms to exist with running times much faster than $O(2^N)$. Since so many solutions are available, we are free to focus on a solution with special properties that makes it easier to find.

Many algorithms for subset sum and subset product are randomized, and hence rely on an assumption about the distribution of the a_i. A little experimentation reveals that elements of $\mathcal{P}$ are not uniformly distributed in $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$, but instead are close to a symmetric distribution (see Section 5).

Definition 1.6. A random variable X on a group G follows a symmetric distribution if for every $a \in G$, $\Pr[X = a] = \Pr[X = a^{-1}]$.

In what follows we give a new algorithm for the random subset product problem that works for instances of high density on groups of the form given in the above construction. Specifically we prove the following two theorems.

Theorem 1.7. *Let $G = G_0$ be an abelian group with subgroups $G_1, G_2, \ldots, G_\ell$ where $|G_i|/|G_{i+1}| = 2^\ell$ for $0 \le i \le \ell - 1$ and $\ell = \sqrt{\log |G|}$. Assume that $a_1, \ldots, a_N$ are independent and distributed symmetrically in G, and that $N > O(4^{\sqrt{\log |G|}} \log |G|)$.*

Then there is an algorithm that solves the subset product problem $(a_1, \ldots, a_N, b)$ with high probability and requires time and space

$$\widetilde{O}\left(4^{\sqrt{\log |G|}}\right) \ .$$

Theorem 1.8. *Let $G = (\mathbb{Z}/\Lambda\mathbb{Z})^\times$ and $\mathcal{P}$ be defined as in the Erdős construction, with the added assumption that $1 \le h_i \le r/i$ for all $1 \le i \le r$. Assume that the elements of $\mathcal{P}$ are independent and distributed symmetrically in G, and that $N = |\mathcal{P}| = K(\Lambda)$. Finally, assume that the probability $p \equiv 1 \bmod Q_i$ for $p \in \mathcal{P}$ is at least $1/(h_i + 1)$ and independent across Q_i.*

Then there is an algorithm that with high probability finds a subset of $\mathcal{P}$ that products to b in G and requires time and space

$$2^{O(\sqrt{(\log N)(\log\log N)^2})} \ .$$

The symbol log will denote the base 2 logarithm, while ln denotes the natural logarithm. Groups are assumed to have efficient implementations of arithmetic.

Thanks to Eric Bach and Carl Pomerance for helpful suggestions. The third and fourth authors are grateful to Mark Liffiton for helpful advice and support regarding both hardware and software.

2. Previous Results

Constructing Carmichael numbers has a long history, one that is ably documented in [9] and [12]. We restrict ourselves to pointing out the more recent results that provide context for our new computations.

The largest tabulation of Carmichael numbers is due to Richard Pinch; his tabulation up to 10^{15} [12] has since been extended to 10^{16}. Alford, Granville, and Pomerance proved there are infinitely many Carmichael numbers in [1]. The authors credit their inspiration to [18], who first used the Erdős heuristic to construct Carmichael numbers with a large number of prime factors. Current records for Carmichael numbers with many prime factors go to Löh and Niebuhr [9] at 1101518 prime factors and an unpublished computation by the first two authors at 19565300 prime factors.

The method of [9] clearly works well in practice, but unfortunately is without a running time analysis. This makes it difficult to fit into the existing subset product literature since it is not clear how the running time depends on N or on Λ. The algorithm exploits the fact that among elements of $\mathcal{P}$, residues of 1 modulo $q_i^{h_i}$ are more common than other residues. It divides b by $p \in \mathcal{P}$ in such a way that the running product has one more residue equal to one at each step, backtracking if necessary. Measuring the closeness of an element of $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$ to the identity is done via Definition 1.2.

There is a large body of literature on the subset sum problem that transfers immediately to the subset product problem. For subset product problems of high density the standard technique is dynamic programming, which in this case would take $O(N\Lambda)$ time and space. Since our goal is to construct Carmichael numbers where N is in excess of 2^{30} and Λ is even bigger, this method is infeasible. A better naive algorithm is to pick a random subset and see if it products to $b \bmod \Lambda$. Even if the elements of $\mathcal{P}$ were distributed uniformly the expected time taken would be $\widetilde{O}(\phi(\Lambda))$. The polynomial time algorithm of [3] is similarly infeasible; with N so large we need an algorithm that is sub-linear in N.

Wagner's algorithm for the k-tree problem [17] has inspired an algorithm for the subset sum problem that gets faster as the density increases [10, 14, 11]. A recent paper by Howgrave-Graham and Joux [7] even gives improvements for most problems of density 1. However, all these methods are exponential time and thus inappropriate for our setting.

Theorem 1.7 was inspired by the Kuperberg sieve from the theory of quantum algorithms [8], which has complexity $2^{O(\sqrt{\log |G|})}$ where elements being matched are in a group G. The same idea of combining elements in pairs to zero out square root of the bit size at each step was presented by Flaxman and Przydatek [5], though they chose to only apply their algorithmic idea to a narrow slice of problems with the proper density to make the algorithm run in polynomial time. A better application to the current setting would be to pick at least $2^{\sqrt{\log \Lambda}}$ elements of $\mathcal{P}$ and pair them up over $\sqrt{\log \Lambda}$ levels, zeroing out $\sqrt{\log \Lambda}$ bits of Λ at each level. The algorithm in Theorem 1.8 does even better by applying the Kuperberg idea to a subgroup of $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$, and the surprise is that there are enough elements of $\mathcal{P}$ that fall in the subgroup so that the algorithm can still succeed.

3. Algorithms

Among the two new subset product algorithms presented in this section, Algorithm 1 (developed by the first two authors) is more appropriate for constructing Carmichael numbers with k prime factors for a variety of k, while Algorithm 2 (developed by the third and fourth authors) is better at constructing Carmichael numbers with a very large number of prime factors. Both build products of primes in $\mathcal{P}$ that get successively close to the identity in $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$, with Algorithm 1 utilizing Definition 1.3 while Algorithm 2 uses Definition 1.2.

The motivation behind Algorithm 1 was the observation by the first author that his implementation of the Erdős heuristic only needed to use a small fraction of the available primes to generate Carmichael numbers. Through repeated runs, each time removing the primes comprising the previous Carmichael number, it produces a set of co-prime Carmichael numbers where the product of any subset also forms a Carmichael number. Because of the inevitable difference in numbers of prime factors, it is likely that the sums of the individual numbers of prime factors will cover a wide range of possibilities.

In order to maximize the chance of getting most of the intermediate numbers of prime factors, we want to generate as many different Carmichael numbers as possible with relatively few prime factors. We achieve this goal over r stages (one for each Q_i), where at stage j we work with a set S_j containing products a with $\bar{\omega}(a) = j$ (we call this set S for simplicity). Let $S_1 = \mathcal{P}$, and let $b_j = \prod_{a \in S_j} a \bmod \Lambda$. For each element in S_j, calculate its residue modulo each of the prime powers dividing Λ. Do the same for each b_j.

First, if $b_j \not\equiv 1 \bmod Q_j$ find and remove the element $e \in S$ that maximizes $\bar{\omega}(e^{-1}b_j)$. As long as there is an element congruent to the product modulo Q_j in stage j (which there will almost certainly be in high density situations), the new product of all primes in S will be 1 modulo Q_i for all $i \leq j$. Then for each remaining element of $a \in S$, we use a greedy algorithm to find $\hat{a} \in S$ maximizing $\bar{\omega}(a \cdot \hat{a})$. For simplicity Algorithm 1 shows $\bar{\omega}(a \cdot \hat{a})$ increasing by only one, but an important improvement is to efficiently find $\hat{a}$ that maximizes the increase to the $\bar{\omega}$ value (see Section 8 for details). At the end of this process, you will have a (potentially empty) set of elements that were not matched. At this point, you can multiply all of these "chaff" together to get an element that is 1 modulo Q_j, as well as being 1 modulo Q_i for $i < j$. Replace S with the set-aside products, along with the product of the chaff, and move on to the next stage.

At the end of the r stages, you have a collection of coprime "base" Carmichael numbers. It is not necessary to compute Carmichael numbers with a particular number of prime factors to prove its existence. Instead, let v be the vector $[1, 0, 0, \ldots, 0]$ of length equal to one greater than the number of total prime factors in the Carmichael numbers. Loop over the base Carmichael numbers. For each number, let v' be v shifted to the right by the number of factors in that Carmichael number. E.g., if the first base Carmichael has 3 factors, $v' = [0, 0, 0, 1, \ldots]$. Let $v = v + v'$. Then, at the end of the loop, the kth position in the vector will represent the number of constructed Carmicheals with $k - 1$ prime factors. (Excepting the first position.)

Our next algorithm constructs a Carmichael number via Step 3b of the Erdős construction. As in Algorithm 1 there will be several running products; the goal is to get these products closer to the identity in $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$. However, this time elements

Algorithm 1: Many Carmichaels subset-product

$S_1 \leftarrow \mathcal{P}$;
for $u \leftarrow 1$ **to** r **do**
  `sort` S_u;
  calculate $b_u = \prod_{a \in S_u} a \bmod Q_u$, remove a from S_u that satisfies $a \equiv b_u \bmod Q_u$;
  for $a \in S_u$ **do**
    Find $\hat{a} \in S_u$ such that $a \cdot \hat{a} \equiv 1 \bmod Q_u$; `/* pushing down */`
    $S_{u+1} \leftarrow a\hat{a}$;
    Remove $a, \hat{a}$ from S ;
  product remaining $a \in S_u$, add to S_{u+1} ;
return S_{r+1}, each element of which is Carmichael ;

are matched for Q_i with i close to r first, and at each level the first element matched is the distinguished product containing b^{-1} modulo Λ. In this way the final identity product is b^{-1} times a number of primes from a subset $\mathcal{T}$ of $\mathcal{P}$, making $\mathcal{P} \setminus \mathcal{T}$ the factors of a Carmichael number.

As discussed above, a heuristic application of the ideas from [5] results in a subset product algorithm that takes time and space $\widetilde{O}(2^{\sqrt{\log \Lambda}})$. However, this is still inefficient since it does not take advantage of the large number of $p \in \mathcal{P}$ with $\omega(p)$ small.

Let N_j be the size of the set $\mathcal{P}_j = \{p \in \mathcal{P} \ : \ \omega(p) \le j\}$. Then define a subgroup G of $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$ as $(\mathbb{Z}/\hat{\Lambda}\mathbb{Z})^\times$ with $\hat{\Lambda} = \prod_{i=1}^m Q_i$, where

$$m = \min_{1 \le j \le r} j \text{ such that } \mathrm{E}[N_j] > (\log \Lambda) 4^{\sqrt{\sum_{i=1}^j h_i \log q_i}} \ .$$

We will see in Section 7 that with reasonable assumptions the expected size of N_j is at least $N \cdot \prod_{i=j+1}^r 1/(h_i+1)$.

For shorthand let $\ell = \sqrt{\log |G|}$. It is important that during the construction of $\mathcal{P}$ we pick out all elements of $\mathcal{P}_m$ and $\Theta(2^{\sqrt{\log |G|}} \log |G|)$ elements of $\mathcal{P}$ with ω values equal to j for $m < j \le r$. The elements with ω value greater than m will be needed to match with b^{-1}, so that a product of primes and b^{-1} has ω value m. This product will be called the distinguished element a_0. The elements of $\mathcal{P}_m$, along with a_0, are then matched over the course of ℓ levels. At each level, products have another ℓ bits eliminated, so that by the end of ℓ levels an identity product has been formed.

Pseudocode is presented as Algorithm 2.

Constructing subgroups of the correct size is not too hard, since $|G|$ will typically be products of small primes to large powers. As an example, note that if $G = (\mathbb{Z}/2^{h_1}\mathbb{Z})^\times$ then $G' = \{a \in G \ : \ a \equiv 1 \bmod 2^e\}$ is a subgroup of G of order $2^{h_1 - e}$ and hence index 2^{e-1}. Thus for this example of G we have $G_0 = G$ and $G_i = \{a \equiv 1 \bmod 2^{i \lfloor \sqrt{|G|} \rfloor}\}$.

Algorithm 2: Large Carmichael subset-product

```
/* Phase 1:  find/construct elements of G                          */
```
Input: set S containing $\mathcal{P}_m$ and $(\log|G|)2^{\sqrt{\log|G|}}$ primes p with $\omega(p) = j$ for each of $m < j \le r$;
$a_0 \leftarrow b^{-1} \bmod \Lambda$; `/* b is product of all elements of P */`
$\mathcal{T} = \emptyset$;
for $u \leftarrow r$ **to** m **do**
  find $p \in \mathcal{P}_u$ such that $a_0 \cdot p \equiv 1 \bmod Q_u$;
  $a_0 \leftarrow a_0 \cdot p \bmod \Lambda$, $\mathcal{T} \leftarrow \mathcal{T} \cup \{p\}$;
add a_0 to S ;
```
/* Phase 2:  continually pair products to reach identity in G  */
```
construct subgroups G_i, $1 \le i \le \ell$ with factor groups having size 2^ℓ ;
for $i \leftarrow 1$ **to** ℓ **do**
  pair the element containing a_0 first to ensure it is included ;
  pair elements of S whose product is in G_i ;
return the history of any element in $G_\ell = \{1_G\}$;

4. Symmetric Distributions

Algorithms 1 and 2 are not guaranteed to succeed. Rather, they will succeed with some positive probability depending upon the distribution of the elements of $\mathcal{P}$. The key result proven in this section is that if the elements of $\mathcal{P}$ are distributed symmetrically, then the probability that the product of two elements is in a subgroup is at least as large as if the elements were distributed uniformly.

First, however, we mention the tail bound that will be used frequently in the analysis that follows. Since products at any level are composed of distinct elements of $\mathcal{P}$, if the initial elements are independent then subsequent products are independent as well.

Theorem 4.1 (Chernoff bound)**.** *Let $X_1, X_2, \ldots, X_n$ be independent Bernoulli trials that take value 1 with probability p_i. Let $X = \sum_{i=1}^{n} X_i$, $\mu = \mathrm{E}[X]$, and δ be any real number in the range $(0, 1]$. Then*

$$\Pr[X < (1-\delta)\mu] < \exp(-\mu\delta^2/2) \ .$$

A classical result is that collision probability is minimized when the distribution is uniform. For a proof see [16, page 66].

Lemma 4.2. *Let X be a random variable on a set S. Then*

$$\sum_{a \in S} \Pr[X = a]^2 \ge \sum_{a \in S} \left(\frac{1}{|S|}\right)^2 \ .$$

Theorems 1.7 and 1.8 assume that the given distribution is symmetric. Our definition is a simple generalization of that in [5].

Definition 4.3. The distribution of a random variable X on G is symmetric if $\Pr[X = a] = \Pr[X = a^{-1}]$ for all $a \in G$. In this case we call X a symmetric random variable.

Recall that all groups under consideration are abelian. If H is a subgroup of G and X is a random variable on G then there is a natural random variable on G/H given by sampling X and then mapping the result to G/H via the map $g \mapsto gH$. Call this random variable X_H. A specific example occurs when $G = (\mathbb{Z}/\Lambda\mathbb{Z})^\times$, we sample X and want the result modulo Q_i. Call this random variable $X \bmod Q_i$. Mapping to G/H preserves the symmetric property.

Proposition 4.4. *Let H be a subgroup of G. If X is symmetric then X_H is symmetric.*

Proof. Let $\phi : G \to G/H$ be the canonical homomorphism. Then

$$\Pr[X_H = \hat{a}H] = \sum_{a \in \hat{a}H} \Pr[X = a] = \sum_{a^{-1} \in \hat{a}^{-1}H} \Pr[X = a] = \sum_{a \in \hat{a}^{-1}H} \Pr[X = a^{-1}]$$

since ϕ a homomorphism implies that $a \in \hat{a}H$ if and only if $a^{-1} \in \hat{a}^{-1}H$. The fact that X is symmetric then yields

$$\sum_{a \in \hat{a}^{-1}H} \Pr[X = a^{-1}] = \sum_{a \in \hat{a}^{-1}H} \Pr[X = a] = \Pr[X_H = \hat{a}^{-1}H] \ .$$

□

Constructing a new group via direct product also preserves the symmetric property for random variables. If X_1, X_2 are independent random variables on H_1, H_2 respectively, then let $X_1 \times X_2$ be a random variable on $G = H_1 \times H_2$. We define this random variable by

$$\Pr[X_1 \times X_2 = (a, b)] = \Pr[X_1 = a] \cdot \Pr[X_2 = b] \ .$$

Proposition 4.5. *If X_1, X_2 are symmetric random variables on H_1, H_2 respectively then $X_1 \times X_2$ is symmetric on $G = H_1 \times H_2$.*

Proof. Follows immediately from the fact that $(a, b)^{-1} = (a^{-1}, b^{-1})$.

□

Products of random variables also preserve the symmetric property.

Proposition 4.6. *Let $X_1, X_2, \ldots, X_n$ be symmetric random variables on G. Then $Y = \prod_{i=1}^n X_i$ is also a symmetric random variable on G.*

Proof. Using the symmetric nature of each of the X_i we have the following identity involving multiple sums.

$$\begin{aligned}
&\Pr[Y = b] \\
&= \sum_{a_1, \ldots a_{n-1} \in G} \Pr[X_1 = a_1] \cdots \Pr[X_{n-1} = a_{n-1}] \Pr[X_n = b \cdot (a_1 a_2 \cdots a_{n-1})^{-1}] \\
&= \sum_{a_1, \ldots a_{n-1} \in G} \Pr[X_1 = a_1^{-1}] \cdots \Pr[X_{n-1} = a_{n-1}^{-1}] \Pr[X_n = b^{-1} \cdot (a_1 a_2 \cdots a_{n-1})] \\
&= \Pr[Y = b^{-1}] \ .
\end{aligned}$$

□

Phase 2 of Algorithm 2 involves matching elements whose product is in some subgroup H of G. Weakening the definition of collision to having $X_1 \cdot X_2$ in a subgroup H yields a similar result to Lemma 4.2: symmetric random variables have a greater than uniform collision probability.

Proposition 4.7. *Let H be a subgroup of G and let X_1, X_2 be independent random variables on G with identical symmetric distributions. Then the probability that $X_1 \cdot X_2$ is in H is at least $|H|/|G|$.*

Proof. Let C be a set of coset representatives of H. By group theory, C has size $|G|/|H|$. $X_i \bmod H$ is symmetric by Proposition 4.4, and thus $\Pr[X_1 \in \hat{a}H] = \Pr[X_2 \in \hat{a}^{-1}H]$ for all $\hat{a}$ in C. Then

$$\begin{aligned}
\Pr[X_1 \cdot X_2 \in H] &= \sum_{a \in G} \Pr[X_1 = a] \Pr[X_2 \in a^{-1}H] \\
&= \sum_{\hat{a} \in C} \sum_{a \in \hat{a}H} \Pr[X_1 = a] \Pr[X_2 \in a^{-1}H] \\
&= \sum_{\hat{a} \in C} \Pr[X_1 \in \hat{a}H] \Pr[X_2 \in \hat{a}^{-1}H] \\
&= \sum_{\hat{a} \in C} \Pr[X_1 \in \hat{a}H]^2 \\
&\geq \sum_{\hat{a} \in C} \left(\frac{1}{|G|/|H|} \right)^2 = \frac{1}{|G|/|H|}
\end{aligned}$$

where the lower bound follows from Lemma 4.2. □

5. Divisors of Λ

Löh and Niebuhr [9] note the distribution of divisors of Λ modulo q_i, but provide no proof. Here we give a full description of the distribution modulo $Q_i = q_i^{h_i}$ in order to provide justification for the claim that elements of $\mathcal{P}$ are distributed symmetrically modulo Λ.

We begin with the following lemma, then extend the distribution to all classes modulo $q_i^{h_i}$.

Lemma 5.1. *Suppose that a divisor d of Λ is chosen uniformly at random from the set of all divisors of Λ. Then*

$$\Pr[d \text{ exactly divisible by } q_i^e] = \frac{1}{h_i + 1}$$

for all $1 \leq i \leq r$ and all $1 \leq e \leq h_i$.

Proof. The divisors of Λ can be identified with r-tuples $(e_1, \ldots, e_r)$, where e_i is power of q_i that divides d. It is thus the case that when the divisors are partitioned by the power of q_i, there are $h_i + 1$ such partitions and they are all of the same size. □

If we now assume that the part of a divisor relatively prime to q_i is uniformly distributed modulo Q_i, then the $1/(h_i{+}1)$ probability of all divisors exactly divisible by q_i^e is shared equally among the $q_i^{h_i - e} - q_i^{h_i - e - 1}$ elements of $\mathbb{Z}/Q_i\mathbb{Z}$ which are exactly divisible by q_i^e. This yields the heuristic distribution

$$\Pr[d \equiv a \bmod q_i^{h_i}] = \begin{cases} \frac{1}{h_i+1} & \text{if } a = 0 \\ \frac{1}{(h_i+1)(q_i^{h_i-e} - q_i^{h_i-e-1})} & \text{if } q_i^e \text{ exactly divides } a \end{cases}$$

It is easy to show that if $d + 1$, $d' + 1$ are multiplicative inverses in $(\mathbb{Z}/Q_i\mathbb{Z})^\times$ and q_i^e exactly divides d then q_i^e exactly divides $d' + 1$. Thus the distribution of

$X = d+1, d \mid \Lambda$ is symmetric on $(\mathbb{Z}/Q_i\mathbb{Z})^\times$, and Proposition 4.5 extends this result to $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$.

A difficult question is whether the distribution remains symmetric with the added condition that $d+1$ be prime. We will assume it does, but it is worth noting that elements of $\mathcal{P}$ do not have the same distribution as divisors of Λ. For example, if $d \equiv kq_i - 1 \bmod q_i^{h_i}$ for some k then $d+1$ is divisible by q_i and thus not prime (note this particular problem does not jeopardize the claim that $\mathcal{P}$ is distributed symmetrically over $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$).

Analysis of Algorithm 2 will depend upon the following heuristic assumptions, which we will henceforth call the standard assumptions.

Heuristic 1 (Standard assumptions)**.** Let X_p be a random variable corresponding to the value modulo Λ of $p \in \mathcal{P}$.

(1) The X_p are symmetric random variables on $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$.
(2) The X_p are independent, as are $X_p \bmod Q_i$ for different $1 \le i \le r$.
(3) The probability that $X_p \equiv 1 \bmod Q_i$ is at least $1/(h_i+1)$.
(4) Λ is constructed so that $1 \le h_i \le r/i$ for all $1 \le i \le r$.
(5) The size of $\mathcal{P}$ is $K(\Lambda)$.

6. Algorithm analysis

The following theorem provides the proof for Theorem 1.7 and is general enough to be applicable to many settings besides constructing Carmichael numbers. We use the notation ℓ for $\sqrt{\log_2 |G|}$, and follow closely the proof of Theorem 3.2 from [8].

Theorem 6.1. *Let $G = G_0$ be an abelian group with subgroups $G_1, G_2, \ldots, G_\ell$ where $|G_i|/|G_{i+1}| = 2^\ell$ for $0 \le i \le \ell - 1$. Suppose that S contains at least $O(\ell^2 4^\ell)$ independent elements of G distributed symmetrically. Then Phase 2 of Algorithm 2 finds a solution to the subset product problem with probability at least $1 - e^{-\Omega(\log |G|)}$ using time and space $\widetilde{O}(4^{\sqrt{\log_2 |G|}})$.*

Proof. We begin by assuming the algorithm has been successful up to level u, so that we have a list L_u of elements in the group G_u. Our goal is to prove by induction that

$$|L_u| \ge C_u \cdot \ell^2 2^{2\ell - u}$$

with high probability, given that $|L_0| = C_0 \ell^2 2^{2\ell}$. Here C_u is defined recursively by $C_0 = 3, C_u = C_{u-1} - 2^{-(\ell-u)}$. For $0 \le u \le \ell$ we have $1 \le C_u \le 3$. As long as $|L_\ell| \ge 1$, then Phase 2 succeeds in finding a solution.

Given $a \in L_u$, let X_b be a Bernoulli random variable that takes value 1 if a and b "match," that is if $a \cdot b \in G_{u+1}$. Then a has a match in L_u as long as $X = \sum_b X_b \ge 1$. Elements of G_u are products of symmetrically distributed elements of S, and thus are symmetrically distributed themselves by Proposition 4.6. It then follows from Proposition 4.7 that $X_b = 1$ with probability at least $|G_u|/|G_{u+1}| = 2^{-\ell}$. Thus $\mathrm{E}[X] \ge \ell^2$ until all but $\ell^2 2^\ell$ elements are matched. If this holds then

$$|L_{u+1}| \ge \frac{|L_u| - \ell^2 2^\ell}{2} \ge \ell^2 2^{2\ell - u - 1} \cdot \left(C_u - 2^{-(\ell-u)} \right) \ .$$

The products are distinct and hence independent, so the Chernoff bound applies. The probability that a fails to match is at most

$$\Pr[X \le \ell^2/2] \le \Pr[X \le (1 - 1/2)\mathrm{E}[X]] \le \exp(-\mathrm{E}[X]/8) \le \exp(-\ell^2/8) \ .$$

We seek to make at most $\ell^2 4^\ell$ matches, so all will succeed with probability at least

$$(1 - e^{-\ell^2/8})^{\ell^2 4^\ell} \geq 1 - \ell^2 4^\ell e^{-\ell^2/8} \ .$$

The probability of all matches succeeding at all ℓ levels is then at least

$$(1 - e^{-\frac{\ell^2}{16}+3\ell})^\ell \geq 1 - \ell e^{-\frac{\ell^2}{16}+3\ell} \geq 1 - e^{-\Omega(\ell^2)} = 1 - e^{-\Omega(\log|G|)} \ .$$

□

7. Bounding the Running Time

The previous section gave a subexponential analysis of Phase 2 running on a general group G. Here we bound $\log|G|$ in terms of N so that the running time of Algorithm 2 can be expressed in terms of the problem size, proving Theorem 1.8. In fact, an easy bound on $\log\Lambda$ would be sufficient asymptotically, but to measure the gain from having G be a subgroup of $(\mathbb{Z}/\Lambda\mathbb{Z})^\times$ we go further and prove that $m = O(\sqrt{r}\log r)$. Throughout we will take the standard assumptions as given.

Our starting point is item (5) of the standard assumptions, namely that

$$|\mathcal{P}| = \frac{\Lambda}{\phi(\Lambda)\ln\sqrt{2\Lambda}} \prod_{i=1}^{r} \left(h_i + \frac{q_i - 2}{q_i - 1}\right)$$

and which we denote by N. Recall that G is defined as the integers modulo $\hat{\Lambda} = \prod_{i=1}^m q_i^{h_i}$ where m is the smallest integer such that N_m is large enough. Here N_m is the number of elements of $\mathcal{P}$ with $\omega(p) \leq m$ and "large enough" means $\mathrm{E}[N_m]$ is large enough for Phase 2 to succeed with high probability. By our standard assumptions the probability that an element of $\mathcal{P}$ is congruent to 1 modulo $q_i^{h_i}$ is at least $\frac{1}{h_i+1}$ and this condition is independent for different values of i. Hence the condition on $\mathrm{E}[N_m]$ becomes

$$\frac{\Lambda}{\phi(\Lambda)\ln\sqrt{2\Lambda}} \prod_{i=1}^{m} \left(h_i + \frac{q_i - 2}{q_i - 1}\right) \prod_{i=m+1}^{r} \frac{h_i + \frac{q_i-2}{q_i-1}}{h_i + 1} > (\log\Lambda) 4^{\sqrt{\sum_{i=1}^m h_i \log q_i}} \ .$$

Clearly $1 \leq m \leq r$ and a smaller m is preferred so we will provide an upper bound. The analysis requires $m > 10$ and $r \geq 64$.

Bounding $\log|G|$ requires several preparatory results. First we prove that $\log\Lambda$ is polynomial in r.

Lemma 7.1. *Assume that $10 < m \leq r$ and that $1 \leq h_i \leq r/i$ for $1 \leq i \leq r$. Then*

$$m < \sum_{i=1}^{m} h_i \log q_i < 2r(\log m)^2 \ .$$

Proof. The q_i are the first r primes and $1 \leq h_i$ for all i. Thus $\sum_{i=1}^m h_i \log q_i > m$.

For the upper bound we require a bound on the mth prime number. From [2, Section 8.8] this is given by $q_m < m(\ln m + \ln\ln m)$ for $m \geq 6$. With $m > 6$ we use the conceptually easier bound of $\log q_m < 2\log m$. Since $h_i \leq r/i$ this yields

$$\sum_{i=1}^{m} h_i \log q_i < 2r \log m \sum_{i=1}^{m} \frac{1}{i} < 2r \log m \cdot (1 + \ln m)$$

and $(1 + \ln m) < \log m$ for $m > 10$. □

An immediate corollary is that $\log\Lambda < 2r(\log r)^2$. Next we find that r is logarithmic in N.

Lemma 7.2. *Assume that $r \geq 64$. Then $\log N > r/3$.*

Proof. We start with the definition of N given above. The term $h_i + \frac{q_i-2}{q_i-1}$ is bounded below by $3/2$ for all $q_i \geq 3$. Since $\Lambda/\phi(\Lambda) > 1$ and $\ln\sqrt{2\Lambda} < \log\Lambda < 2r(\log r)^2$ by Lemma 7.1 we have

$$N = \frac{\Lambda}{\phi(\Lambda)\ln\sqrt{2\Lambda}} \prod_{i=1}^{r} \left(h_i + \frac{q_i-2}{q_i-1}\right) > \frac{1}{2r(\log r)^2}\left(\frac{3}{2}\right)^{r-1}$$

Thus $\log N > (r-1)\log(3/2) - (1 + \log r + 2\log\log r) > r/3$ when $r \geq 64$. □

When bounding $\mathrm{E}[N_m]$ a sticky term is $\prod(h_i+1)/(h_i + \frac{q_i-2}{q_i-1})$. It ought to be close to one; we provide a bound logarithmic in r.

Lemma 7.3. *Assume $r \geq 16$. Then for all $m \geq 1$*

$$\prod_{j=m+1}^{r} \frac{h_j+1}{h_j + \frac{q_j-2}{q_j-1}} < (\ln r)^2 \ .$$

Proof. Start with the transformation

$$\frac{h_j+1}{h_j+\frac{q_j-2}{q_j-1}} = \frac{h_j + \frac{q_j-2}{q_j-1} + 1 - \frac{q_j-2}{q_j-1}}{h_j+\frac{q_j-2}{q_j-1}} = 1 + \frac{1/(q_j-1)}{h_j+\frac{q_j-2}{q_j-1}} = 1 + \frac{1}{(q_j-1)h_j + q_j - 2} \ .$$

By assumption $q_j \geq 3$ and $h_j \geq 1$, so $(q_j-1)h_j \geq 2$ for all $2 \leq j \leq r$, giving the bound

$$1 + \frac{1}{(q_j-1)h_j + q_j - 2} \leq 1 + \frac{1}{q_j} \ .$$

We saw in Lemma 7.1 that $r \geq 6$ implies $q_r < 2r\ln r$. We now use a result from [2, Section 8.8] on the prime reciprocal sum, namely

$$\sum_{j=1}^{r} \frac{1}{q_j} \leq \sum_{q < 2r\ln r} \frac{1}{q} < \ln\ln(2r\ln r) + B + \frac{1}{(\ln(2r\ln r))^2} \ .$$

where the sums are over primes and $B < 0.27$ is the prime-reciprocal constant. This upper bound is less than $2\ln\ln r$ as long as $r \geq 16$. The slope of the tangent line to $y = e^x$ at $x = 0$ is 1, which means $e^x \geq 1 + x$ for all positive x. With $x = 1/q_j$ this gives

$$\prod_{j=m+1}^{r}\left(1+\frac{1}{q_j}\right) < \exp\left(\sum_{j=1}^{r}\frac{1}{q_j}\right) < \exp(2\ln\ln r) = (\ln r)^2 \ .$$

□

With this preparatory work out of the way the bound on $\log|G|$ follows from properly bounding $\mathrm{E}[N_m]$, the expected number of elements of $\mathcal{P}$ with ω-value m.

Lemma 7.4. *Assume r and N are sufficiently large. Given the standard assumptions,*

$$\log|G| < 27\log N(\log\log N)^2 \ .$$

Proof. Recall that G is defined as the group of units modulo $\prod_{i=1}^{m} q_i^{h_i}$ where m is the smallest integer such that $N\prod_{i=m+1}^{r}\frac{1}{h_i+1} > (\log\Lambda)4^{\sqrt{\log|G|}}$. Since m is the smallest such integer, multiplying by $1/(h_m+1)$ flips the inequality. Thus

$$\frac{1}{h_m+1}\frac{\Lambda}{\phi(\Lambda)\ln\sqrt{2\Lambda}}\prod_{i=1}^{m}\left(h_i+\frac{q_i-2}{q_i-1}\right)\prod_{i=m+1}^{r}\frac{h_i+\frac{q_i-2}{q_i-1}}{h_i+1} < (\log\Lambda)4^{\sqrt{\log|G|}} \ .$$

Focusing on the left hand side, $h_m < r$ by construction, $\Lambda/\phi(\Lambda) > 1$, $\ln\sqrt{2\Lambda} < \log\Lambda < 2r(\log r)^2$ by Lemma 7.1, and $\prod(h_i+\frac{q_i-2}{q_i-1})(h_i+1) > (\ln r)^{-2}$ by Lemma 7.3. Combining all these bounds yields

$$\frac{1}{r+1}\frac{1}{2r(\log r)^2}\frac{1}{(\ln r)^2}\prod_{i=1}^{m}\left(h_i+\frac{q_i-2}{q_i-1}\right) < (\log\Lambda)4^{\sqrt{\log|G|}}$$

$$\implies \sum_{i=1}^{m}\log\left(h_i+\frac{q_i-2}{q_i-1}\right)-(2\log 2r+4\log\log r) < \log\log\Lambda+2\sqrt{\sum_{i=1}^{m}h_i\log q_i}$$

$$\implies m\log(3/2)-3\log r < \log\log\Lambda+2\sqrt{2r(\log m)^2}$$

where $\sum_{i=1}^{m}h_i\log q_i < 2r(\log m)^2$ follows from Lemma 7.1.

If $m > 6\sqrt{r}\log r$ then $m\log(3/2) > 2\sqrt{2r}\log m+\log\log\Lambda+3\log r$ for sufficiently large r so we must have $m < 6\sqrt{r}\log r$.

Since $\log|G| < 2r(\log m)^2$ this bound on m gives us $\log|G| < 2r(\log 6+\frac{1}{2}\log r+\log\log r)^2$ which is at most $2r(\frac{3}{2}\log r)^2$ if $r \geq 32$. Lemma 7.2 now completes the proof.

□

Although asymptotically $\log|G|$ and $\log\Lambda$ are equivalent at $O(\log N(\log\log N)^2)$, this work showing $m = O(\sqrt{r}\log r)$ provides a theoretical justification for the gains seen in practice. We now prove Theorem 1.8, restated here for convenience. It was proven that $m < 6\sqrt{r}\log r$ in Lemma 7.4 so the necessary assumption that $m > 2\sqrt{r}\log r$ causes no harm.

Theorem 7.5. *Let $G = (\mathbb{Z}/\Lambda\mathbb{Z})^\times$ and $\mathcal{P}$ be defined as in the Erdős construction, with the added assumption that $1 \leq h_i \leq r/i$ for all $1 \leq i \leq r$. Assume that the elements of $\mathcal{P}$ are independent and distributed symmetrically in G, and that $N = |\mathcal{P}| = K(\Lambda)$. Finally, assume that the probability $p \equiv 1 \bmod Q_i$ for $p \in \mathcal{P}$ is at least $1/(h_i+1)$ and independent across Q_i.*

Then there is an algorithm that with probability at least $1-e^{-\Omega(\log\Lambda)}$ finds a subset of $\mathcal{P}$ that products to b in G and requires time and space

$$2^{O(\sqrt{(\log N)(\log\log N)^2})} \ .$$

Proof. First assume a solution is found and that it includes a_0, the distinguished element of G. Then $b^{-1} \bmod \Lambda$ times some product of primes is the identity in G, and since each a_i is the identity in $(\mathbb{Z}/\Lambda\mathbb{Z})^\times/G$, we have discovered a set of primes in $\mathcal{P}$ that product to b modulo Λ.

In bounding the probability of failure we focus first on the size of $\mathcal{P}_m$ and the distinguished element a_0. We notated $|\mathcal{P}_m|$ by N_m, and chose G so that

$$\mathrm{E}[N_m] > (\log\Lambda)4^{\sqrt{\log|G|}} \ .$$

For $p \in \mathcal{P}$ let X_p be a Bernouilli random variable that takes value 1 if $\omega(p) = m$, and let X be the sum of all X_p. Then $\mathrm{E}[X] = \mathrm{E}[N_m]$ and by the Chernoff bound

$$\Pr[X < (1 - 1/2)\mathrm{E}[N_m]] < \exp(-\mathrm{E}[N_m]/8)$$
$$\implies \Pr\left[X < \frac{1}{2}(\log \Lambda)4^{\sqrt{\log |G|}}\right] < \exp\left(-\frac{\log \Lambda}{8}\right) \ .$$

The distinguished element is dealt with in a different fashion. For levels $u = r$ to $m+1$ we multiply b by an element p such that $\omega(p) = u$ and $p \equiv b \bmod q_u^{h_u}$. Focus on level u and let Y_p be a Bernoulli random variable that is 1 if it satisfies that condition, with Y being the sum over all Y_p. The worst case is at level $m+1$, where

$$\Pr[Y_p = 1] \geq \frac{1}{q_{m+1}^{h_{m+1}} - q_{m+1}^{h_{m+1}-1}} \prod_{i=m+2}^{r} \frac{1}{h_i + 1} \geq \frac{1}{q_{m+1}^{h_{m+1}}} \prod_{i=m+2}^{r} \frac{1}{h_i + 1}$$

and hence

$$\mathrm{E}[Y] \geq \frac{N}{q_{m+1}^{h_{m+1}}} \prod_{i=m+2}^{r} \frac{1}{h_i + 1} \geq \frac{1}{q_{m+1}^{h_m+1}} \cdot \mathrm{E}[N_m] \geq \frac{4^{\sqrt{\log |G|}}}{q_{m+1}^{h_m+1}} \cdot \log \Lambda \ .$$

We will show $(h_{m+1} \log q_{m+1})^2 < 4 \sum_{i=1}^{m} h_i \log q_i$ and hence that $q_{m+1}^{h_{m+1}} < 4^{\sqrt{\log |G|}}$. Since the h_i are non-increasing, $4 \sum_{i=1}^{m} h_i \log q_i \geq 4m h_{m+1}$. Meanwhile,

$$h_{m+1}(\log q_{m+1})^2 \leq \frac{r}{m+1} \cdot (2 \log (m+1))^2 \leq 4m$$

since $m > 2\sqrt{r} \log r$ implies $m^2 + m \geq 4r(\log r)^2$.

Now the Chernoff bound can again be applied, giving

$$\Pr[Y < (1 - 1/2)(\log \Lambda)] \leq \exp\left(-\log \Lambda/8\right) \ .$$

This is the worst case among the at most r levels, which means the total success probability of Phase 1 is at least

$$(1 - e^{-\Omega(\log \Lambda)})(1 - e^{-\Omega(\log \Lambda)})^r \geq 1 - (r+1)e^{-\Omega(\log \Lambda)} > 1 - (3 \log N + 1)e^{-\Omega(\log \Lambda)}$$

where $r < 3 \log N$ follows by Lemma 7.2.

With S containing $O((\log \Lambda)4^{\sqrt{\log |G|}})$ elements, Theorem 6.1 allows us to conclude that Phase 2 completes with probability at least $1 - e^{-\Omega(\log \Lambda)}$. Note that the distinguished element is distributed symmetrically by Proposition 4.6 and that Theorem 6.1 assumes it is always the first to be matched. The total probability of success is thus also $1 - e^{-\Omega(\log \Lambda)}$.

Finally we note the running time. We start the algorithm with at most $O(r(\log \Lambda)4^{\sqrt{\log |G|}})$ elements in S. Finding the right product to put b^{-1} in G takes r searches at a cost of $O(\sqrt{\log |G|})$ each. Phase 2 takes time and space $\widetilde{O}((\log \Lambda)4^{\sqrt{\log |G|}})$. Applying Lemma 7.4 then gives a total resource usage of $2^{O(\sqrt{(\log N)(\log \log N)^2})}$. □

8. Construction

Algorithm 1 was implemented in Python and run on a T3E. The best result so far is that Carmichael numbers have been constructed with k prime factors for every k between 3 and 19565220.

It is important to detail how one can efficiently "push down" several primes in one step. Let a be an element of the set S at stage j. We seek $\hat{a}$ that maximizes

TABLE 1. Large Carmichael numbers

$\Lambda = 2^{15}\cdot 3^8\cdot 5^5\cdot 7^4\cdot 11^3\cdot 13^2\cdot 17^2\cdot 19^2\cdot 23^2\cdot 29\cdot 31\cdot 37\cdot 41\cdot 43\cdot 47\cdot 53\cdot 59\cdot 61\cdot 67\cdot 71\cdot 73\cdot 79$
$= 288828494392627542423975683172283292832395366400000$
$k = 1021449117$, $\lvert\mathcal{P}\rvert = 1021449926$, $K(\Lambda) = 1009441849$
$n = \ldots 428427434862714073557504000001$, $\lvert\mathcal{T}\rvert = 809$, $d = 25564327391$
Subset Product Time = 174 sec
$\Lambda = 2^{16}\cdot 3^7\cdot 5^5\cdot 7^4\cdot 11^3\cdot 13^2\cdot 17^2\cdot 19^2\cdot 23^2\cdot 29^2\cdot 31\cdot 37\cdot 41\cdot 43\cdot 47\cdot 53\cdot 59\cdot 61\cdot 67\cdot 71\cdot 73\cdot 79\cdot 83\cdot 89\cdot 97$
$= 400116635717624630133804016619530416834808037326786560000$
$k = 10333229505, \lvert\mathcal{P}\rvert = 10333230324, K(\Lambda) = 10225023621$
$n = \ldots 795363156023129401777459200001$, $\lvert\mathcal{T}\rvert = 819$, $d = 295486761801$
Subset Product Time = 98 sec

$\bar{\omega}(a\cdot\hat{a})$. For a set A let $\bar{A} = \{a^{-1} \bmod \Lambda : \ a \in A\}$. Then combine the sets $S' = \{a \ : \ a \in S \text{ and } a < a^{-1} \bmod \Lambda\}$ and $\overline{S \setminus S'}$ and sort lexicographically on the list of residues. Then for our element a, the element of $\overline{S \setminus S'}$ closest to a in the $\bar{\omega}$ metric will be the element immediately following a or preceding a. We use S' rather than combining S and $\bar{S}$ because in the latter case we end up with duplicates. Once you find the associated $\hat{a}$, if the maximum possible value of $\bar{\omega}(a\cdot\hat{a})$ at the jth stage is j, proceed to the next element. If it is greater than j, then remove both a and b from S, multiply and set aside.

Algorithm 2 was implemented using C++ and NTL [15] and run on a 3.3 GHz processor with 16 GB of main memory. Two Carmichael numbers are presented in Table 1. Here, k is the number of prime factors of n and d denotes the number of decimal digits of n, while $\mathcal{T}$ is the set output by Algorithm 2 and removed. As conjectured in [9], $K(\Lambda)$ is within 3% of $|\mathcal{P}|$. The last thirty digits of each Carmichael number are also included. The billion prime case took more time because the method of calculating G resulted in Phase 2 operating on 27.7 million primes, as opposed to 16.5 million for the ten billion factor case. This neatly demonstrates how the number of primes needed for Phase 2 grows more slowly than N.

For Algorithm 2 the most important implementation detail was the instantiation of elements of $\mathcal{P}$. Since such primes have the property that $p-1$ divides Λ, one can store only the exponents of the divisors of Λ, thereby fitting primes into a single 64-bit `long` for all cases under consideration.

However, one cannot multiply elements of $\mathcal{P}$ and maintain the property that one less than the element divides Λ. We created a class `ModElement` which encapsulates an integer modulo Λ, a vector of condensed elements of $\mathcal{P}$ that product to the integer (called the "history"), and methods that product such elements or compute its ω value. In this way, when some element is the identity, the history of that element gives the solution to the subset product problem.

We implemented the subgroups G_i as integers modulo M_i where M_i is an appropriate divisor of Λ. Ease of implementation made this an attractive alternative to a generic group model, but one disadvantage is that M_i is sometimes larger than the order of the subgroup it represents. For example, if 5^2 divides Λ and 5 divides M_i, then M_{i+1} will be divisible by 5^2 in order that products at the next level be congruent to 1 mod 25, and hence be five times too big.

A significant improvement to Algorithm 2 comes from passing elements congruent to 1 modulo M_i to the next level without matching. Since elements congruent to 1 modulo q_i^e are more common, observed sizes of L_i are larger than $L_{i-1}/2$.

As for the underlying data structure, the necessary requirements are that one have constant time insertion and removals, and that one can quickly find elements with a particular value modulo M_i. Our solution was a hash table of `ModElements` keyed to the residues modulo M_i of the value.

Even though this data structure makes each prime expensive to store, the total memory needed is quite manageable due to the relatively small number of primes needed out of the total. For example, in the ten billion case we use only about 16.5 million primes, since $G = (\mathbb{Z}/\hat{\Lambda}\mathbb{Z})^\times$ where $\hat{\Lambda} = 2^{16}\cdot 3^7\cdot 5^5\cdot 7^4\cdot 11^3\cdot 13^2\cdot 17^2\cdot 19^2\cdot 23^2\cdot 29^2\cdot 31\cdot 37\cdot 41\cdot 43\cdot 47\cdot 53\cdot 59$.

9. Future Work

In [6] the authors extend the basic Erdős construction to a variety of other pseudoprimes. Most likely the methods in this paper can be extended as well.

References

1. W. R. Alford, Andrew Granville, and Carl Pomerance, *There are infinitely many Carmichael numbers*, Ann. of Math. (2) **139** (1994), no. 3, 703–722.
2. Eric Bach and Jeffrey Shallit, *Algorithmic number theory. Vol. 1*, Foundations of Computing Series, MIT Press, Cambridge, MA, 1996, Efficient algorithms.
3. Mark Chaimovich, *New algorithm for dense subset-sum problem*, Astérisque (1999), no. 258, xvi, 363–373, Structure theory of set addition.
4. P. Erdős, *On pseudoprimes and Carmichael numbers*, Publ. Math. Debrecen **4** (1956), 201–206.
5. Abraham D. Flaxman and Bartosz Przydatek, *Solving medium-density subset sum problems in expected polynomial time*, STACS 2005, Lecture Notes in Comput. Sci., vol. 3404, Springer, Berlin, 2005, pp. 305–314.
6. Dominique Guillaume and François Morain, *Building pseudoprimes with a large number of prime factors*, Appl. Algebra Engrg. Comm. Comput. **7** (1996), no. 4, 263–277.
7. Nick Howgrave-Graham and Antoine Joux, *New generic algorithms for hard knapsacks*, Advances in cryptology—EUROCRYPT 2010, Lecture Notes in Comput. Sci., vol. 6110, Springer, Berlin, 2010, pp. 235–256.
8. Greg Kuperberg, *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem*, SIAM J. Comput. **35** (2005), no. 1, 170–188 (electronic).
9. Günter Löh and Wolfgang Niebuhr, *A new algorithm for constructing large Carmichael numbers*, Math. Comp. **65** (1996), no. 214, 823–836.
10. Vadim Lyubashevsky, *The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem*, APPROX-RANDOM, Lecture Notes in Comput. Sci., vol. 3624, Springer, 2005, pp. 378–389.
11. Lorenz Minder and Alistair Sinclair, *The extended k-tree algorithm*, SODA '09: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (Philadelphia, PA, USA), Society for Industrial and Applied Mathematics, 2009, pp. 586–595.
12. R. G. E. Pinch, *The Carmichael numbers up to* 10^{15}, Math. Comp. **61** (1993), no. 203, 381–391.
13. Srinivasa Ramanujan, *Highly composite numbers*, Ramanujan J. **1** (1997), no. 2, 119–153, Annotated and with a foreword by Jean-Louis Nicolas and Guy Robin.
14. Andrew Shallue, *An improved multi-set algorithm for the dense subset sum problem*, Algorithmic number theory, Lecture Notes in Comput. Sci., vol. 5011, Springer, Berlin, 2008, pp. 416–429.
15. Victor Shoup, *Number theory library (NTL)*, `http://www.shoup.net/ntl`.
16. Abraham Sinkov, *Elementary cryptanalysis, a mathematical approach*, first ed., Random House New Mathematical Library, vol. 22, Random House, New York, 1968.

17. David Wagner, *A generalized birthday problem (extended abstract)*, Advances in Cryptology – CRYPTO 2002, Lecture Notes in Comput. Sci., vol. 2442, Springer, Berlin, 2002, pp. 288 – 303.
18. Ming Zhi Zhang, *A method for finding large Carmichael numbers*, Sichuan Daxue Xuebao **29** (1992), no. 4, 472–479.

Institute for Defense Analyses, Center for Computing Sciences, 17100 Science Drive, Bowie, MD 20715, United States
E-mail address: grantham@super.org

Illinois Wesleyan University, 1312 Park St, Bloomington, IL 61701, United States
E-mail address: shayman@iwu.edu

Illinois Wesleyan University, 1312 Park St, Bloomington, IL 61701, United States
E-mail address: ashallue@iwu.edu